第 2 章

程序语言基础知识

2.1 备考指南

2.1.1 考纲要求

根据考试大纲中相应的考核要求，在“**程序设计语言基础知识**”模块中，要求考生掌握以下方面的内容。

- 汇编、编译、解释系统的基础知识和基本工作原理
- 程序设计语言的基本成分(数据、运算、控制和传输)，程序调用的实现机制
- 各类程序设计语言的主要特点和适用情况

2.1.2 考点统计

“**程序语言基础知识**”模块，在历次软件设计师考试试卷中出现的考核知识点及分值分布情况如表 2.1 所示。

表 2.1 历次考点统计表

年 份	题 号	知 识 点	分 值
2017 年下半年	上午：20～22、48～50	程序设计语言、脚本语言的概念、编辑器和解释器、语法分析、DFA 的状态转换图、函数调用	6 分
	下午：无	无	0 分
2017 年上半年	上午：20～22、48～50	程序设计语言的基本成分、正规式、编译过程、DFA 的状态转换图、函数调用、表达式语法树	6 分
	下午：无	无	0 分

续表

年　份	题　号	知 识 点	分　值
2016年下半年	上午：20～22、48～50	逻辑表达式、函数参数的传递、数组、正规式、上下文无关文法、语义分析	6分
	下午：无	无	0分
2016年上半年	上午：20～22、48～50	编译和解释方式的不同、脚本语言的概念、中间代码表示形式、移进—归约分析法、有限自动机、传址调用和传值调用	6分
	下午：无	无	0分
2015年下半年	上午：20～22、48～50	编译器和解释器、逆波兰式、动态语义错误、NFA的状态转换图、函数调用	6分
	下午：无	无	0分
2015年上半年	上午：20～22、48～50	程序设计语言的基本成分、表达式树、变量的存储、解释程序、NFA到DFA的转换、自顶向下的语法分析方法	6分
	下午：无	无	0分
2014年下半年	上午：18、21～22、48～50	程序设计语言、后缀式、中间代码、编译过程、有限自动机、上下文无关文法	6分
	下午：无	无	0分
2014年上半年	上午：20～22、48～50	程序设计语言、函数参数传递、编译过程、编译与解释的区别、上下文无关文法、语法分析	6分
	下午：无	无	0分

2.1.3　命题特点

纵观历次试卷，本章知识点是以选择题的形式出现在试卷中的。在历次考试上午试卷中，所考查的题量大约为6道选择题，所占分值为6分(约占试卷总分值75分中的8%)。本章试题主要考查考生是否掌握了相关的理论知识，难度中等。有限自动机是本章的重点，也是难点。

2.2　考点串讲

2.2.1　程序语言概述

一、程序设计语言的基本概念

1．低级语言和高级语言

1) 低级语言

通常称机器语言和汇编语言为低级语言。机器语言是指用0、1字符串组成的机器指令序列，是最基本的计算机语言；汇编语言是指用符号表示指令的语言。

2) 高级语言

高级语言是从人类的逻辑思维角度出发、面向各类应用的程序语言，其抽象程度大大

提高，需要编译成特定机器上的目标代码才能执行。这类语言与人们使用的自然语言比较接近，大大提高了程序设计的效率。

2．编译程序和解释程序

用某种高级语言或汇编语言编写的程序称为源程序，源程序不能直接在计算机上执行。如果源程序是使用汇编语言编写的，则需要一个称为汇编程序的翻译程序将其翻译成目标程序后才能执行。如果源程序是使用某种高级语言编写的，则需要相应的解释程序或编译程序对其进行翻译，然后才能在机器上运行。

3．程序设计语言的定义

程序设计语言的定义一般都涉及语法、语义、语用和语境等方面。

(1) 语法：由程序设计语言的基本符号组成程序中的各个语法成分(包括程序)的一组规则，其中由基本字符构成的符号(单词)书写规则称为词法规则，由符号(单词)构成语法成分的规则称为语法规则。程序语言的语法可通过形式语言进行描述。

(2) 语义：程序语言中按语法规则构成的各个语法成分的含义，可分为静态语义和动态语义。

(3) 语用：表示构成语言的各个记号和使用者的关系，涉及符号的来源、使用和影响。

(4) 语境：理解和实现程序设计语言的环境，包括编译环境和运行环境。

4．程序设计语言的分类

1) 命令式程序设计语言

命令式程序设计语言是基于动作的语言，在这种语言中，计算被看作是动作的序列。命令式语言族开始于 Fortran、Pascal 和 C 语言，体现了命令式程序设计的关键思想。

2) 面向对象的程序设计语言

面向对象的程序设计在很大程度上应归功于从模拟领域发展而来的 Simula，Simula 提出了对象和类的概念。C++、Java 和 Smalltalk 是面向对象程序设计语言的代表。

3) 函数式程序设计语言

函数式程序设计语言是一类以 λ-演算为基础的语言。该语言的代表是 LISP，其中大量使用了递归。

4) 逻辑型程序设计语言

逻辑型程序设计语言是一类以形式逻辑为基础的语言。该语言的代表是建立在关系理论和一阶谓词理论基础上的 Prolog。

二、程序设计语言的基本成分

1．数据成分

程序语言的数据成分是指一种程序语言的数据类型。

1) 常量和变量

按照程序运行时数据的值能否改变，可将数据分为常量和变量。程序中的数据对象可以具有左值和(或)右值，左值是指存储单元(或地址、容器)，右值是指具体值(或内容)。变量具有左值和右值，在程序运行过程中其右值可以改变；常量只有右值，在程序运行过程中其右值不能改变。

2) 全局量和局部量

按照数据的作用域范围，可将数据分为全局量和局部量。系统为全局量分配的存储空间在程序运行的过程中一般是不改变的，而为局部量分配的存储单元是动态改变的。

3) 数据类型

按照组织形式的不同，可将数据分为基本类型、用户定义类型、构造类型及其他类型，等等。C(C++)的数据类型如下。

- 基本类型：整型(int)、字符型(char)、实型(float、double)和布尔类型(bool)。
- 特殊类型：空类型(void)。
- 用户定义类型：枚举类型(enum)。
- 构造类型：数组、结构和联合。
- 指针类型：type *。
- 抽象数据类型：类类型。

其中，布尔类型和类类型是C++在C语言的基础上扩充的。

2．运算成分

程序语言的运算成分是指允许使用的运算符号及运算规则。大多数高级程序语言的基本运算可以分成算术运算、关系运算和逻辑运算，有些语言还提供位运算。运算符号的使用与数据类型密切相关。为了确保运算结果的唯一性，运算符号要规定优先级和结合性，必要时还要使用圆括号。

3．控制成分

控制成分指明语言允许表述的控制结构，程序员使用控制成分来构造程序中的控制逻辑。

1) 顺序结构

在顺序结构中，计算过程从所描述的第一个操作开始，按顺序依次执行后续的操作，直到执行完序列的最后一个操作。顺序结构内也可以包含其他控制结构。

2) 选择结构

选择结构提供了在两种或多种分支中选择执行其中一个分支的逻辑。基本的选择结构是指定一个条件P，然后根据条件的成立与否决定控制流走计算A还是走计算B，从两个分支中选择一个执行。选择结构中的计算A或计算B还可以包含顺序、选择和循环结构。程序语言中通常还提供简化了的选择结构，也就是没有计算B的分支结构。

3) 循环结构

循环结构描述了重复计算的过程，通常包括3个部分：初始化、需要重复计算的部分和重复的条件。其中，初始化部分有时在控制的逻辑结构中不进行显式的表示。循环结构主要有两种形式：while型循环结构和do-while型循环结构。

4．C(C++)语言提供的控制语句

C(C++)语言提供的控制语句如下。

(1) 复合语句。复合语句用于描述顺序控制结构。复合语句是一系列用“{”和“}”括起来的声明和语句，其主要作用是将多条语句组成一个可执行单元。复合语句是一个整体，要么全部执行，要么一条语句也不执行。

(2) if语句和switch语句。这两种语句用于实现选择结构。

① if 语句实现的是双分支的选择结构，其一般形式如下：

```
if(表达式)语句 l;else 语句 2;
```

其中，语句 1 和语句 2 可以是任何合法的 C(C++)语句，当语句 2 为空语句时，可以简化为

```
if(表达式) 语句;
```

使用 if 语句时，需要注意的是 if 和 else 的匹配关系。C 语言规定，else 总是与离它最近的尚没有 else 与其匹配的 if 相匹配。

② switch 语句描述了多分支的选择结构，其一般形式如下：

```
switch(表达式){
    case 常量表达式 1: 语句 1;
    case 常量表达式 2: 语句 2;
    …
    case 常量表达式 n: 语句 n;
    default: 语句 n+1;
}
```

执行 switch 语句时，首先计算表达式的值，然后用所得的值与列举的常量表达式值依次比较，若任一常量表达式都不能与所得的值相匹配，则执行 default 的“语句 n+l”，然后结束 switch 语句。

表达式可以是任何类型，常用的是字符型或整型表达式。多个常量表达式可以共用一个语句组。语句组可以包括任何可执行语句，且无须用“{”和“}”括起来。

(3) 循环语句。C(C++)语言提供了 3 种形式的循环语句用于描述循环计算的控制结构。

① while 语句。while 语句描述了先判断条件再执行循环体的控制结构，其一般形式为

```
while(条件表达式) 循环体语句;
```

② do-while 语句。do-while 语句描述了先执行循环体再判断条件的控制结构，其一般形式为

```
do
    循环体语句;
while(条件表达式);
```

③ for 语句。for 语句的基本形式为

```
for(表达式 1;表达式 2;表达式 3)循环体语句;
```

5．函 数

函数是程序模块的主要成分，它是一段具有独立功能的程序。函数的使用涉及 3 个概念：函数定义、函数声明和函数调用。

(1) 函数定义：包括函数首部和函数体两个部分。函数定义描述了函数做什么和怎么做。

(2) 函数声明：函数应该先声明后引用。函数声明定义了函数原型。声明函数原型的目的在于告诉编译器传递给函数的参数个数、类型以及函数返回值的类型，参数表中仅需要依次列出函数定义中参数的类型。函数原型可以使编译器检查源程序中对函数的调用是否正确。

(3) 函数调用：当需要在一个函数(称为主调函数)中使用另一个函数(称为被调函数)实现的功能时，便以函数名字进行调用，称为函数调用。主调函数和被调函数之间交换信息的方法主要有两种：一种是由被调用函数把返回值返回给主调函数，另一种是通过参数带回信息。函数调用时，实际参数(简称实参)和形式参数(简称形参)间交换信息的方法有传值调用和引用调用两种。

- 传值调用(Call by Value)。若实现函数调用时实参向形参传递相应类型的值(副本)，则称为传值调用。这种方式下形参不能向实参传递信息。在 C 语言中，要实现被调用函数对实参的修改，必须用指针作形参。即调用时需要先对实参进行取地址运算，然后将实参的地址传递给指针形参，本质上仍属于传值调用。这种方式实现了间接内存访问。
- 引用调用(Call by Reference)。引用是 C++中增加的数据类型，当形参为引用类型时，形参名实际上是实参的别名，函数中对形参的访问和修改实际上就是针对相应实参所做的访问和改变。

2.2.2 语言处理程序基础

一、汇编语言基本原理

1．汇编语言

汇编语言是为特定的计算机或计算机系统设计的面向机器的符号化的程序设计语言。用汇编语言编写的程序称为汇编语言源程序。汇编语言源程序由若干条语句组成。一个程序中可以有 3 类语句：指令语句、伪指令语句和宏指令语句。

(1) 指令语句：又称为机器指令语句，汇编后能产生相应的机器代码，被 CPU 直接识别并执行相应的操作。指令语句可分为传送指令、算术运算指令、逻辑运算指令、移位指令、转移指令和处理机控制指令等。

(2) 伪指令语句：指示汇编程序在对源程序进行汇编时完成某些工作。其与指令语句的区别是，伪指令语句经汇编后不产生机器代码；另外，伪指令语句所指示的操作是在源程序被汇编时完成的，而指令语句的操作必须在程序运行时完成。

(3) 宏指令语句：将多次重复使用的程序段定义为宏。宏的定义必须按照相应的规定进行，每个宏都有相应的宏名。

2．汇编程序

汇编程序的功能是将用汇编语言编写的源程序翻译成机器指令程序。它一般至少需要两次扫描源程序才能完成翻译过程。第一次扫描的主要工作是定义符号的值并创建一个符号表(ST)，另外，有一个固定的机器指令表 MOT1，其中记录了每条机器指令的记忆码和指令的长度；第二次扫描的任务是产生目标程序，除了使用前一次扫描所产生的符号表(ST)外，还要使用机器指令表 MOT2。在第二次扫描过程中，可执行汇编语句应被翻译成对应的二进制代码机器指令。这一过程涉及两个方面的工作：一是把机器指令助记符转换成二进制机器指令操作码，这可通过查找 MOT2 来实现；二是求出操作数区各操作的值(用二进制表示)。

二、编译程序基本原理

编译程序的功能是把用高级语言书写的源程序翻译成与之等价的目标程序。编译过程划分成词法分析、语法分析、语义分析、中间代码生成、代码优化和目标代码生成六个阶段，实际的编译器可能会将其中的某些阶段结合在一起进行处理。

1) 词法分析阶段

词法分析阶段的任务是对源程序从前到后(从左到右)逐个字符进行扫描，从中识别出一个个“单词”符号。“单词”符号是程序设计语言的基本语法单位，如关键字、标识符等。词法分析程序输出的“单词”常常采用二元组的方式，即单词类别和单词自身的值。

2) 语法分析阶段

语法分析的任务是在词法分析的基础上，根据语言的语法规则将单词符号序列分解成各类语法单位，如“表达式”“语句”和“程序”等。词法分析和语法分析本质上都是对源程序的结构进行分析。

过程(函数)说明和过程(函数)调用是程序中一种常见的语法结构。过程说明和调用语句的翻译，有赖于形式参数和实际参数结合的方式以及数据空间的分配方式。需要分配存储空间的对象有基本数据类型、结构化数据类型和连接数据(如返回地址、参数等)。分配的依据是名字的作用域和生存期的定义规则。分配的策略有静态存储分配和动态存储分配两大类。

- 如果在编译时就能确定目标程序运行时所需要的全部空间大小，则在编译时就安排好目标程序运行时的全部数据空间，并确定每个数据对象的存储位置。这种分配策略为静态存储分配。
- 如果一个程序语言允许递归过程和可变数据结构，那么就需要采用动态存储分配技术。动态存储分配策略的实现有栈分配和堆分配两种方式。

3) 语义分析阶段

语义分析阶段主要是审查源程序是否存在语义错误，并收集类型信息供后面的代码生成阶段使用，只有语法和语义都正确的源程序才能翻译成正确的目标代码。语义分析的一个主要工作是进行类型分析和检查。

描述程序语义的形式化方法主要有属性文法、公理语义、操作语义和指称语义等，其中属性文法是对上下文无关文法的扩充。目前广泛使用的静态语义分析方法是语法制导翻译，其基本思想是将语言结构的语义以属性的形式赋予代表此结构的文法符号；而属性的计算以语义规则的形式赋予文法的产生式。在语法分析的推导或归纳的步骤中，通过语义规则实现对属性的计算，以达到对语义的处理。

4) 中间代码生成阶段

中间代码是一种结构简单且含义明确的记号系统，可以有多种形式。中间代码生成阶段的工作就是根据语义分析的输出生成中间代码。语义分析和中间代码生成所依据的是语言的语义规则。

实际上，中间代码起着编译器前端和后端分水岭的作用，使用中间代码有利于提高编译程序的可移植性。常用的中间代码有后缀式、三元式、四元式和树等形式。

- 后缀式：把运算符写在运算对象的后面。例如，a*b 的后缀式为 ab*。这种表示法的优点是根据运算对象和运算符的出现次序进行计算，不需要使用括号，也便于用栈实现求值。

- 三元式：由运算符 OP、第一运算对象 ARG1 和第二运算对象 ARG2 组成。例如，x:=a+b 的三元式为：① (+, a, b) ② (:=, ①, x)。
- 四元式：组成成分为运算符 OP、第一运算对象 ARG1、第二运算对象 ARG2 和运算结果 RESULT。例如，x:=a+b 的四元式为：① (+, a, b, t) ② (:=, t, _, x)。

5) 代码优化阶段

代码优化阶段的任务是对前一阶段产生的中间代码进行变换或进行改造，目的是使生成的目标代码更为高效，即省时间和省空间。优化过程可以在中间代码生成阶段进行，也可以在目标代码生成阶段进行。

6) 目标代码生成阶段

目标代码生成阶段的任务是把中间代码变换成特定机器上的绝对指令代码、可重定位的指令代码或汇编指令代码。这是编译的最后阶段，它的工作与具体的机器密切相关。

7) 符号表管理

符号表管理阶段的任务是在符号表记录源程序中各个符号的必要信息，以辅助语义的正确性检查和代码生成。符号表的建立可以始于词法分析阶段，也可以放到语法分析阶段，但符号表的使用有时会延续到目标代码的运行阶段。

8) 出错处理

用户编写的源程序中的错误大致可分为静态错误和动态错误。动态错误也称动态语义错误，指程序中包含的逻辑错误。静态错误是指编译阶段发现的程序错误，可分为语法错误和静态语义错误。出错处理程序的任务包括检查错误、报告出错信息、排错、恢复编译工作。

三、解释程序的基本原理

解释程序是一种语言处理程序，在词法、语法和语义分析方面与编译程序的工作原理基本相同，但在运行用户程序时，它直接执行源程序或源程序的内部形式(中间代码)。因此，解释程序并不产生目标程序，这是它和编译程序的主要区别。

解释程序的结构通常可以分成两部分：第一部分是分析部分，包括通常的词法分析、语法分析和语义分析程序，经语义分析后把源程序翻译成中间代码，中间代码常采用逆波兰表示形式；第二部分是解释部分，用来对第一部分产生的中间代码进行解释执行。

2.2.3 文法和有限自动机

一、文法和语言的形式描述

1．文法的定义

描述语言语法结构的形式规则称为文法。文法 G 是一个四元组，可表示为 $G=(V_N, V_T, P, S)$。其中，V_T 是一个非空有限集，其每个元素称为一个终结符；V_N 是一个非空有限集，其每个元素称为一个非终结符。$V_N \cap V_T = \phi$。P 是产生式的有限集合，每个产生式都是形如 $\alpha \rightarrow \beta$ 的规则，其中 α 称为产生式的左部，β 称为产生式的右部。$S \in V_N$，称为开始符号，它至少要在一条产生式中作为左部出现。

2．文法的分类

乔姆斯基把文法分成四种类型，即 0 型、1 型、2 型和 3 型。

- 0 型文法也称为短语文法，其能力相当于图灵机。
- 1 型文法也称为上下文有关文法，这种文法意味着对非终结符的替换必须考虑上下文，并且一般不允许替换成 ε 串，此文法对应于线性有界自动机。
- 2 型文法也称为上下文无关文法，对非终结符的替换无须考虑上下文，它对应于下推自动机。
- 3 型文法等价于正规式，因此也称为正规文法或线性文法，它对应于有限状态自动机。

3．句子和语言

设有文法 $G=(V_N, V_T, P, S)$，则有如下概念。

- 推导和直接推导：从文法的开始符号 S 出发，反复使用产生式，将产生式左部的非终结符替换为右部的文法符号序列，直至产生一个终结符的序列为止。若有产生式 $\alpha \to \beta \in P$、$\gamma, \delta \in V^*$，则 $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$ 称为文法 G 中的一个直接推导。
- 直接归约和归约：若文法 G 中有一个直接推导 $\alpha \Rightarrow \beta$，则称 α 是 β 的一个直接归约；若文法 G 中有一个推导 $\gamma \overset{*}{\underset{G}{\Rightarrow}} \delta$，则称 γ 是 δ 的一个归约。
- 句型和句子：若文法 G 的开始符号为 S，那么，从开始符号 S 能推导出的符号串称为文法的一个句型，即 α 是文法 G 的一个句型，当且仅当有如下推导 $S \overset{*}{\underset{G}{\Rightarrow}} \alpha$，$\alpha \in V^*$。若 X 是文法 G 的一个句型，且 $X \in V_T^*$，则称 X 是文法 G 的一个句子。
- 语言：从文法 G 的开始符号出发，所能推导出的句子的全体称为文法 G 产生的语言，记为 $L(G)$。

4．文法的等价

若文法 G_1 与文法 G_2 产生的语言是相同的，即 $L(G_1)=L(G_2)$，则称这两个文法是等价的。

二、词法分析

1．正规表达式和正规集

对于字母表 $\sum$，其上的正规式及其表示的正规集可以递归定义如下。

(1) ε 是一个正规式，它表示集合 $L(\varepsilon)=\{\varepsilon\}$。

(2) 若 a 是 $\sum$ 上的字符，则 a 是一个正规式，它所表示的正规集为 $\{a\}$。

(3) 若正规式 r 和 s 分别表示正规集 $L(r)$ 和 $L(s)$，则

- $r|s$ 是正规式，表示集合 $L(r) \cup L(s)$。
- $r \cdot s$ 是正规式，表示集合 $L(r)L(s)$。
- r^* 是正规式，表示集合 $(L(r))^*$。
- (r) 是正规式，表示集合 $L(r)$。

仅由有限次地使用上述三个步骤定义的表达式才是 $\sum$ 上的正规式，其中运算符“|”“·”“*”分别称为“或”“连接”和“闭包”。若两个正规式表示的正规集相同，则认为两者

等价。

2．有限自动机

有限自动机是一种识别装置的抽象概念，它能够正确地识别正规集。

1) 确定的有限自动机

一个确定的有限自动机(DFA)是个五元组：$(S, \sum, f, s_0, Z)$。其中：

- S 是一个有限集，其每个元素称为一个状态。
- $\sum$ 是一个有限字母表，其每个元素称为一个输入字符。
- f 是 $S\times\sum\to S$ 上的单值部分映像。
- $s_0\in S$ 是唯一的一个开始状态。
- Z 是非空的终止状态集合，$Z\subseteq S$。

一个 DFA 可以用两种直观的方式表示：状态转换图和状态转换矩阵。状态转换图简称为转换图，它是一个有向图。DFA 中的每个状态对应转换图中的一个节点，DFA 中的每个转换函数对应图中的一条有向弧，若转换函数为 $f(A,a)=Q$，则该有向弧从节点 A 出发，进入节点 Q，字符 a 是弧上的标记。状态转换矩阵可以用一个二维数组 M 表示，矩阵元素的行下标表示状态，列下标表示输入字符，$M[A,a]$的值是当前状态为 A、输入为 a 时，应转换到的下一状态。在转换矩阵中，一般以第一行的行下标所对应的状态作为初态，而终态则需要特别指出。

2) 不确定的有限自动机

一个不确定的有限自动机(NFA)也是一个五元组，它与确定的有限自动机的区别如下。

- f 是 $S\times\sum\to 2^S$ 上的映像。对于 S 中的一个给定状态及输入符号，返回一个状态的集合。
- 有向弧上的标记可以是 ε。

显然，DFA 是 NFA 的特例。

实际上，对于每个 NFA M，都存在一个 DFA N，且 $L(M)=L(N)$。

对于任何两个有限自动机 M_1 和 M_2，如果 $L(M_1)=L(M_2)$，则称 M_1 和 M_2 是等价的。

3．NFA 到 DFA 的转换

设 NFA $N=(S, \sum, f, s_0, Z)$，与之等价的 DFA $M=(S', \sum, f', q_0, Z')$，用子集法将非确定的有限自动机确定化的算法步骤如下。

(1) 求出 DFA M 的初态 q_0，此时 S'仅含初态 q_0，并且没有标记。

(2) 对于 S'中尚未标记的状态 $q_i=\{s_{i1}, s_{i2}, \text{L}, s_{im}\}$和 $s_{ij}\in S(j=1,2,\text{L},m)$进行如下处理。

① 标记 q_i。

② 对于每个 $a\in\sum$，令 $T=f(s_{i1}, s_{i2}, \text{L}, s_{im}, a)$，$q_j=\varepsilon\_\text{CLOSURE}(T)$。

③ 若 q_i 尚不在 S'中，则将 q_j 作为一个未加标记的新状态添加到 S'，并把状态转换函数 $f'(q_i,a)=q_j$ 添加到 DFA M。

(3) 重复步骤(2)，直到 S'中不再有未标记的状态为止。

(4) 令 $Z'=\{q\,|\,q\in S'\text{且}q\,\text{I}\ Z\neq\phi\}$。

注：若 I 是 NFA N 的状态集合的一个子集，其中 $\varepsilon\_CLOSURE(I)$ 的定义如下。

① 状态集 I 的 $\varepsilon\_CLOSURE(I)$ 是一个状态集。

② 状态集 I 的所有状态属于 $\varepsilon\_CLOSURE(I)$。

③ 若 s 在 I 中，那么从 s 出发经过任意条 ε 弧到达的状态 s' 都属于 $\varepsilon\_CLOSURE(I)$。

从 NFA 转换得到的 DFA 不一定是最简化的，可以通过等价变换将 DFA 进行最小化处理。

三、正规式与有限自动机之间的转换

(1) 对于 $\sum$ 上的 NFA M，可以构造一个 $\sum$ 上的正规式 R，使得 $L(R)=L(M)$。

构造过程分两步进行。

① 在 M 的状态转换图中加两个节点 x 和 y。

② 按图 2.1 所示的方法逐步消去 M 中除 x 和 y 的所有节点。

(2) 对于 $\sum$ 上的每一个正规式 R，可以构造一个 $\sum$ 上的 NFA M，使得 $L(M)=L(R)$。

构造过程分两步进行。

① 对于正规式 R，可用图 2.2 所示的拓广状态图表示。

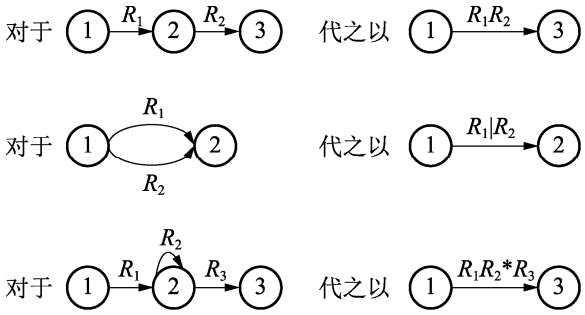


图 2.1　状态转换图(消去中间节点)

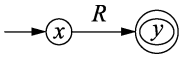


图 2.2　拓广状态图

② 通过对正规式 R 进行分裂并加入新的节点，逐步把图转变成每条弧上的标记是 $\sum$ 上的一个字符或 ε，转换规则如图 2.3 所示。

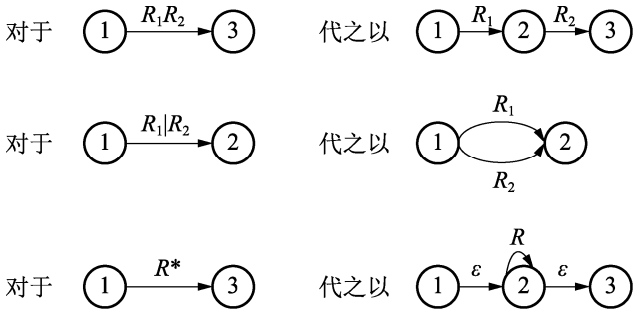


图 2.3　状态转换图(加入新节点)

四、词法分析器的构造

词法分析器的构造过程如下。

(1) 用正规式描述语言中的单词构成规则。

(2) 为每个正规式构造一个 NFA，用于识别正规式所表示的正规集。
(3) 将构造出的 NFA 转换成等价的 DFA。
(4) 对 DFA 进行最小化处理，使其最简。
(5) 根据 DFA 构造词法分析器。

五、语法分析

语法分析的任务是根据语言的语法规则，分析单词串是否构成短语和句子，同时检查和处理程序中的语法错误。根据产生语法树的方向，语法分析可分为自顶向下和自底向上两类。

所谓自顶向下的分析，是对给定的符号串试图自顶向下地为其构造一棵语法树，或者说从文法的开始符号出发，为其构造一个最左推导。

所谓自底向上的分析，是对给定的符号串试图自底向上地为其构造一棵语法树，或者说从给定的符号串本身出发，试图将其归约为文法的开始符号。

算符优先文法属于自底向上的分析法，它利用各个算符间的优先关系和结合规则来进行语法分析，特别适用于分析各种表达式。算符优先文法的任何产生式的右部都会出现两个非终结符相邻的情况，且任何一对终结符之间至多只有 3 种算符关系“>”“<”和“=”之一成立。

2.3 真题详解

综合知识试题

试题 1 (2017 年下半年上午试题 20)

以下更适合用来开发操作系统的编程语言是 (20) 。

(20) A．C/C++　　B．Java　　C．Python　　D．JavaScript

答案：A

解析：除了 C/C++语言以外，其他语言都不具备某些特性、不够方便或软件效率不高。开发操作系统需要编程语言提供以下几个特征：

- 跨平台，不能是只在某个平台下编译。
- 必须是编译型语言，或者有一个非常高效的解释器。
- 必须有方便的操作硬件的功能，容易嵌入汇编。
- 兼容性要好，最好不同编译器编译的符号基本相同，容易链接。
- 编译器本身最好是由该语言自己完成的(大部分语言的编译器都是用 C/C++写的)。
- 开发者可以很方便地扩展、改造或者使用第三方的运行库。
- 开发者众多。
- 该语言开发操作系统的资料要足够完善。

试题 2 (2017 年下半年上午试题 21)

以下关于程序设计语言的叙述中，不正确的是 (21) 。

(21) A．脚本语言中不使用变量和函数
　　B．标记语言常用于描述格式化和链接
　　C．脚本语言采用解释方式实现
　　D．编译型语言的执行效率更高

答案：A

解析：脚本语言可以使用变量和函数。

试题 3　(2017 年下半年上午试题 22)

将高级语言源程序通过编译或解释方式进行翻译时，可以先生成与源程序等价的某种中间代码。以下关于中间代码的叙述中，正确的是__(22)__。

(22) A．中间代码常采用符号表来表示
　　B．后缀式和三地址码是常用的中间代码
　　C．对中间代码进行优化要依据运行程序的机器特性
　　D．中间代码不能跨平台

答案：A

解析：中间代码是源程序的不同表示形式，或称中间语言、中间表示。中间代码表示形式有不同层次、目的之分，如：

- AST(Abstract Syntax Tree，抽象语法树)。
- TAC(Three-Adress Code，三地址码或四元式)。
- P_code(用于 Pascal 语言实现)。
- Bytecode(Java 编译器的输出，Java 虚拟机的输入)。
- SSA(Static Single Assignment Form，静态单赋值形式)。

中间代码还有如下特性：

生成中间代码时，可以不考虑机器的特性，编写生成中间代码的编译程序相对容易。

由于中间代码与具体机器无关，能将生成中间代码的编译程序方便移植到其他机器上，只需要为中间代码开发一个解释器或者将中间代码翻译成目标指令就能在目标机上运行。

在中间代码上更便于做优化处理，某些优化方法在中间代码上比在汇编码后机器代码上更容易实施。

试题 4　(2017 年下半年上午试题 48)

编译过程中进行的语法分析主要是分析__(48)__。

(48) A．源程序中的标识符是否合法　　B．程序语句的含义是否合法
　　C．程序语句的结构是否合法　　D．表达式的类型是否合法

答案：C

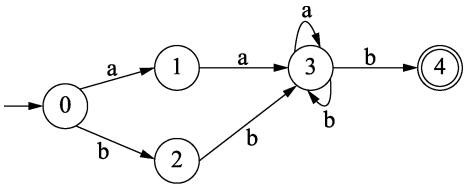
解析：词法分析阶段是编译过程的第一个阶段。这个阶段的任务是从左到右一个字符一个字符地读入源程序，即对构成源程序的字符流进行扫描然后根据构词规则识别单词(也称单词符号或符号)。词法分析程序实现这个任务。

语法分析是编译过程的一个逻辑阶段。语法分析的任务是在词法分析的基础上将单词序列组合成各类语法短语，如“程序”“语句”“表达式”等。语法分析程序判断源程序在结构上是否正确。

语义分析是编译过程的一个逻辑阶段。语义分析的任务是对结构上正确的源程序进行上下文有关性质的审查，进行类型审查。

试题5 (2017年下半年上午试题49)

某确定的有限自动机(DFA)的状态转换如下图所示(0是初态，4是终态)，则该DFA能识别 (49) 。



(49) A．aaab　　B．abab　　C．bbba　　D．abba

答案：(49) A

解析：略。

试题6 (2017年下半年上午试题50)

函数main()、f()的定义如下所示。调用函数f()时，第一个参数采用传值(call by value)方式，第二个参数采用传引用(call by referen)方式，则函数main()执行后输出的值为 (50) 。

main()

```
int x = 10;
f(x,x);
print(x);
```

f(int x, int &a)

```
x = 2*x – 1;
a = a + x;
return;
```

(50) A．10　　B．19　　C．20　　D．29

答案：D

解析：x= 2*10−1=19，a=a+x=10+19=29，a是引用传递，所以结果为29。

试题7 (2017年上半年上午试题20)

在高级语言源程序中，常需要用户定义的标识符为程序中的对象命名，常见的命名对象有 (20) 。

①关键字(或保留字)　②变量　③函数　④数据类型　⑤注释

(20) A．①②③　　B．②③④　　C．①③⑤　　D．②④⑤

答案：B

解析：关键字和注释不能作为标识符给对象命名。

试题8 (2017年上半年上午试题21)

在仅由字符a、b构成的所有字符串中，其中以b结尾的字符串集合可用正规式表示为 (21) 。

(21) A．(b|ab)*b　　B．(ab*)*b　　C．a*b*b　　D．(a|b)*b

答案：D

解析：正规式(a|b)*对应的正规集为{ ε，a，b，aa，ab，…，所有由a和b组成的字符

串}，结尾为 b。

试题 9　(2017 年上半年上午试题 22)

在以阶段划分的编译过程中，判断程序语句的形式是否正确属于__(22)__阶段的工作。

(22) A．词法分析

B．语法分析

C．语义分析

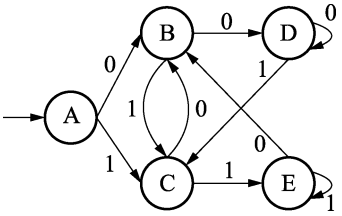
D．代码生成

答案：B

解析：检查单个词是否正确，属于词法阶段的工作。而识别判断程序语句形式是否正确属于语法分析的工作。

试题 10　(2017 年上半年上午试题 48)

某确定的有限自动机(DFA)的状态转换如下图所示(A 是初态，D、E 是终态)，则该 DFA 能识别__(48)__。



(48) A．00110　　B．10101　　C．11100　　D．11001

答案：C

解析：只有 C 选项的字符串能被 DFA 解析。解析路径为 ACEEBDD。

试题 11　(2017 年上半年上午试题 49)

函数 main()、f()的定义如下所示，调用函数 f()时，第一个参数采用传值 (call by value)方式，第二个参数采用传引用(call by reference)方式，main()函数中的 print(x)执行后输出的值为__(49)__。

main()

```
int x = 5;
f(x+1,x);
print(x);
```

f(int x, int &a)

```
x = x*x – 1;
a = x + a;
return;
```

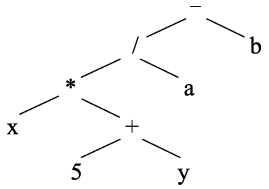
(49) A．11　　B．40　　C．45　　D．70

答案：B

解析：当值传递的时候，将原来的参数复制了一份，但是引用传递的时候是将变量本身传了出去，因此，a 代表的其实就是 x 本身，f 函数里面的 x 是另一个变量，只有 a 的变化才能导致 main 函数里面的 x 值的变化。

试题 12　(2017 年上半年上午试题 50)

下图为一个表达式的语法树，该表达式的后缀形式为__(50)__。



(50) A．x 5 y + * a / b -　　B．x 5 y a b*+/-

C．-/ * x + 5 y a b　　D．x 5 * y + a/b-

答案：A

解析：要得到题目中的表达式语法树的后缀形式，只需要对树进行后序遍历，后序遍历的结果为 x5y+*a/b-。

试题 13 (2016 年下半年上午试题 20)

逻辑表达式求值时常采用短路计算方式。“&&”“||”“！”分别表示逻辑与、或、非运算，“&&”“||”为左结合，“！”为右结合，优先级从高到低为“！”“&&”“||”。对逻辑表达式“x&&(y ||!z)”进行短路计算方式求值时，__(20)__。

(20) A．x 为真，则整个表达式的值即为真，不需要计算 y 和 z 的值

B．x 为假，则整个表达式的值即为假，不需要计算 y 和 z 的值

C．x 为真，再根据 z 的值决定是否需要计算 y 的值

D．x 为假，再根据 y 的值决定是否需要计算 z 的值

答案：B

解析：在进行逻辑与“&&”运算时，只有当两个操作数的值为真，最后的结果才会为真。因此一旦 x 的值为假，整个运算表达式的值则为假。

试题 14 (2016 年下半年上午试题 21)

常用的函数参数传递方式有传值与传引用两种，__(21)__。

(21) A．在传值方式下，形参与实参之间互相传值

B．在传值方式下，实参不能是变量

C．在传引用方式下，修改形参实质上改变了实参的值

D．在传引用方式下，实参可以是任意的变量和表达式

答案：C

解析：传值调用最显著的特征就是被调用的函数内部对形参的修改不影响实参的值。引用调用是将实参的地址传递给形参，使得形参的地址就是实参的地址。

试题 15 (2016 年下半年上午试题 22)

二维数组 a[1..N，1..N]可以按行存储或按列存储。对于数组元素 a[i,j](1<=i,j<=N)，当__(22)__时，在按行和按列两种存储方式下，其偏移量相同。

(22) A．i≠j　　B．i=j　　C．i>j　　D．i<j

答案：B

解析：当 i=j 时得到一个行数与列数相同的二位数组，故偏移量相同。

试题 16　(2016 年下半年上午试题 48)

由字符 a、b 构成的字符串中，若每个 a 后至少跟一个 b，则该字符串集合可用正规式表示为__(48)__。

(48) A．(b|ab)*　　B．(ab*)*　　C．(a*b*)*　　D．(a|b)*

答案：A

解析：本题考查程序语言知识。

正规式(b|ab)表示正规集为{b,ab}，{b|ab}*表示的正规集为{ε,b,ab,bb,bab,abb,abab,bbb,bbab,babb,babab,abbb,abbab,ababb,ababab,…}，用自然语言描述就是每个 a 后面至少有 1 个 b。

正规式(ab*)表示的正规集为{ε,a,ab,abb,abbb,abbbb,…}，(ab*)*表示的正规集为{aa,aab,aabb,aabbb,aabbbb,aba,abba,abbba,abab,abbab,…}，用自然语言描述就是除了空串，每个串中都至少有 1 个 a。

正规式(a*b*)*和(a|b)*是等价的，它们都表示{ε,a,b,aa,ab,ba,bb,aaa,aab,aba,abb,baa,bab,bab,bbb,…}，用自然语言描述就是用 a、b 构成的任何字符串。

试题 17　(2016 年下半年上午试题 49)

乔姆斯基(Chomsky)将文法分为 4 种类型，程序设计语言的大多数语法现象可用其中的__(49)__描述。

(49) A．上下文有关文法　　B．上下文无关文法

C．正规文法　　D．短语结构文法

答案：B

解析：上下文无关文法是形式语言理论中一种重要的变换文法，用来描述上下文无关语言，在乔姆斯基分层中称为 2 型文法。由于程序设计语言的语法基本上都是上下文无关文法，因此应用十分广泛。

试题 18　(2016 年下半年上午试题 50)

运行下面的 C 程序代码段，会出现__(50)__错误。

```
int k=0;
for( ; k<100; );
{k++;}
```

(50) A．变量未定义　B．静态语义　　C．语法　　D．动态语义

答案：D

解析：在本题中，for 语句后有“;”号，说明该循环语句的语句体为空，此时，循环会是一个死循环，所以存在语义错误。

在 C 语言中，直接写{}用于产生独立作用域。编译能通过，符合词法、语法、静态语义。

试题 19　(2016 年上半年上午试题 20)

以下关于高级程序设计语言实现的编译和解释方式的叙述中，正确的是__(20)__。

(20) A．编译程序不参与用户程序的运行控制，而解释程序则参与

B．编译程序可以用高级语言编写，而解释程序只能用汇编语言编写

C．编译方式处理源程序时不进行优化，而解释方式则进行优化

D．编译方式不生成源程序的目标程序，而解释方式则生成

答案：A

解析：编译程序的功能是把用高级语言书写的源程序翻译成与之等价的目标程序。编译过程划分成词法分析、语法分析、语义分析、中间代码生成、代码优化和目标代码生成6个阶段。目标程序可以独立于源程序运行。

解释程序是一种语言处理程序，在词法、语法和语义分析方面与编译程序的工作原理基本相同，但在运行用户程序时，它是直接执行源程序或源程序的内部形式(中间代码)。因此，解释程序并不产生目标程序，这是它和编译程序的主要区别。

试题20 (2016年上半年上午试题21)

以下关于脚本语言的叙述中，正确的是__(21)__。

(21) A．脚本语言是通用的程序设计语言

B．脚本语言更适合应用在系统级程序开发中

C．脚本语言主要采用解释方式实现

D．脚本语言中不能定义函数和调用函数

答案：C

解析：脚本语言是为了缩短传统的编写—编译—链接—运行(edit-compile-link-run)过程而创建的计算机编程语言。脚本语言是一种解释性的语言，例如 Python、JavaScript、ActionScript 等。它不像 C/C++等可以编译成二进制代码，以可执行文件的形式存在；脚本语言不需要编译，可以直接用，由解释器负责解释。

试题21 (2016年上半年上午试题22)

将高级语言源程序先转化为一种中间代码是现代编译器的常见处理方式。常用的中间代码有后缀式、__(22)__、树等。

(22) A．前缀码　　B．三地址码　　C．符号表　　D．补码和移码

答案：B

解析：几种常用的中间代码表示形式为后缀式、树形式、三元式、四元式。

试题22 (2016年上半年上午试题48)

移进—归约分析法是编译程序(或解释程序)对高级语言源程序进行语法分析的一种方法，属于__(48)__的语法分析方法。

(48) A．自顶向下(或自上而下)　　B．自底向上(或自下而上)

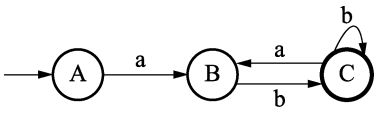
C．自左向右　　D．自右向左

答案：B

解析：移进—归约分析是一种自底向上的分析方法。它的思想是对输入符号串自左至右进行扫描，并将输入符逐个移入一个栈中，当栈顶符号串形成某个句型的句柄时，就用该产生式的左部非终结符代替相应右部的文法符号串，这称为一步规约。重复这一过程，直到规约到栈中只剩文法的开始符号时，则为分析成功。

试题 23　(2016 年上半年上午试题 49)

某确定的有限自动机(DFA)的状态转换如下图所示(A 是初态，C 是终态)，则该 DFA 能识别 (49) 。



(49) A．aabb　　B．abab　　C．baba　　D．abba

答案：B

解析：初态为 A，终态为 C，初态时输入 a 时转换到下一状态 B；要转换到状态 C，可在状态 B 时输入 b，或者在状态 C 时输入 b。

试题 24　(2016 年上半年上午试题 50)

函数 main()、f()的定义如下所示，调用函数 f()时，第一个参数采用传值(call by value)方式，第二个参数采用传引用(call by reference)方式，main 函数中的 print(x)执行后输出的值为 (50) 。

main()

```
int x = 1;
f(5,x);
print(x);
```

f(int x, int &a)

```
x = 2*x + 1;
a = a + x;
return;
```

(50) A．1　　B．6　　C．11　　D．12

答案：D

解析：传址调用，在函数中的操作会改变传入的参数值，而传值调用则不会。调用函数 f()时，第一个参数采用传值方式，5 传给了形参 x；第二个参数采用传引用方式，a 的地址即为实参 x 的地址，因此对 a 的改变也就是对实参 x 的改变。在函数中，x=2*x+1=2*5+1=11，a=a+x=1+11=12，执行后 x 的值变为 12，因此输出 12。

试题 25　(2015 年下半年上午试题 20~21)

编译器和解释器是两种基本的高级语言处理程序。编译器对高级语言源程序的处理过程可以划分为词法分析、语法分析、语义分析、中间代码生成、代码优化、目标代码生成等阶段，其中，(20) 并不是每个编译器都必需的，与编译器相比，解释器 (21) 。

(20) A．词法分析和语法分析　　B．语义分析和中间代码生成
　　C．中间代码生成和代码优化　　D．代码优化和目标代码生成

(21) A．不参与运行控制，程序执行的速度慢
　　B．参与运行控制，程序执行的速度慢
　　C．参与运行控制，程序执行的速度快
　　D．不参与运行控制，程序执行的速度快

答案：C　B

解析：在编译过程中中间代码的生成与优化不是必需的，但使用中间代码有很多好处，最重要的是两点：①便于实现优化，使最终代码的质量更高；②通过中间代码实现前后级分离，在多系统、多语言开发时，可大幅提高整体开发效率，减少开发成本，缩短开发周

期。所以实际的编译系统多数都会使用中间代码。

在解释器上运行程序比直接运行编译过的代码要慢，是因为解释器每次都必须去分析并转译它所运行的程序行，而编译过的程序直接运行即可。

试题26 (2015年下半年上午试题22)

表达式采用逆波兰式表示时，利用__(22)__进行求值。

(22) A．栈　　B．队列　　C．符号表　　D．散列表

答案：A

解析：逆波兰将运算符写在操作数之后，因此也称为后缀表达式。将一个普通的中序表达式转换为逆波兰表达式的一般过程是：从左至右扫描表达式，如果当前字符为变量或者数字，则入栈；如果是运算符，则将栈顶两个元素弹出做相应运算，结果再入栈；以此类推，最后当表达式扫描完，栈里的就是结果。

试题27 (2015年下半年上午试题48)

某程序运行时陷入死循环，则可能的原因是程序中存在__(48)__。

(48) A．词法错误　　B．语法错误

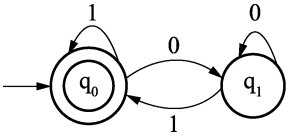
C．动态的语义错误　　D．静态的语义错误

答案：C

解析：死循环错误属于典型的语义错误，但静态的语义错误可被编译器发现，若程序真正陷入死循环说明编译器并未发现，所以属于动态语义错误。

试题28 (2015年下半年上午试题49)

某非确定的有限自动机(NFA)的状态转换如下图所示(q_0既是初态也是终态)。以下关于该NFA的叙述中，正确的是__(49)__。



(49) A．其可识别的0、1序列的长度为偶数

B．其可识别的0、1序列中0与1的个数相同

C．其可识别的非空0、1序列中开头和结尾字符都是0

D．其可识别的非空0、1序列中结尾字符是1

答案：D

解析：在初态q_0输入1又回到q_0，输入0则迁移到状态q_1，可见在0之前可以有任意多个1(1*)；在q_1状态输入0，则回到q_1，输入1则迁移到终态q_0，因此在1之前可以有任意多个0(0*)，因此可识别的字符串为1*00*1。

试题29 (2015年下半年上午试题50)

函数t()、f()的定义如下所示，若调用函数t时传递给x的值为5，并且调用函数f()时，第一个参数采用传值(call by value)方式，第二个参数采用引用(call by reference)方式，则函数t的返回值为__(50)__。

main()

```
int a;
a = 3*x + 1;
f(x,a);
return a-x;
```

f(int x, int &a)

```
int x;
x = 2*s + 1; s = x + r
r=x − 1;
return;
```

(50) A．33　　B．22　　C．11　　D．负数

答案：A

解析：在函数 t 中，执行语句 a=3*x+1，得 a=16；调用 f(x,a)时，将 x 的值 5、a 的值 16 传递给函数 f 的形参 r 和 s。由于 r 采用的是传值方式，函数调用后不会改变 x 的值；而参数 s 采用的是引用方式，函数调用后 a 的值发生改变。函数 f 执行完成后 a 的值变为 38，x 的值不变，为 5，因此函数 t 的返回值为 a−x=38−5=33。

试题 30　(2015 年上半年上午试题 20)

以下关于程序设计语言的叙述中，错误的是__(20)__。

(20) A．程序设计语言的基本成分包括数据、运算、控制和传输等

B．高级程序设计语言不依赖于具体的机器硬件

C．程序中局部变量的值在运行时不能改变

D．程序中常量的值在运行时不能改变

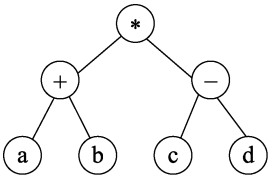
答案：C

解析：变量具有左值和右值，在程序运行过程中，局部变量的右值可以改变。

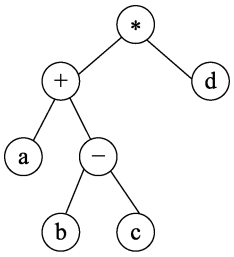
试题 31　(2015 年上半年上午试题 21)

与算术表达式“(a+(b-c))*d”对应的树是__(21)__。

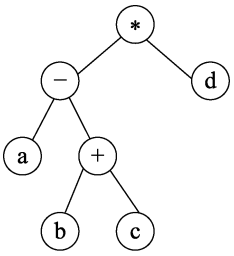
(21) A.



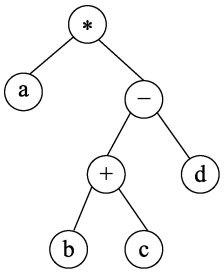
B.



C.



D.



答案：B

解析：表达式用树形来表示时通常采用中缀形式，运算符在树中放在非终端结点的位置，操作数放在叶子结点处。处理时，首先找到运算级别最低的运算符“*”作为根结点，

继而确定该根结点的左、右子树结点在表达式串中的范围为 a+(b-c)和 d；再在对应的范围内寻找运算级别最低的运算符作为子树的根结点，直到范围内无运算符，则剩余的变量或数为表达式树的叶子。

试题32 (2015年上半年上午试题22)

C 程序中全局变量的存储空间在__(22)__分配。

(22) A．代码区　　B．静态数据区　C．栈区　　D．堆区

答案：B

解析：代码区：存放函数体的二进制代码。

栈区：由编译器自动分配释放，存放函数的参数值、局部变量的值等。

堆区：一般由程序员分配释放，若程序员不释放，程序结束时可能由操纵系统回收。

静态数据区：内存在程序启动的时候才被分配，而且可能直到程序开始执行的时候才被初始化，所分配的内存在程序的整个运行期间都存在，如全局变量、static 变量等。

试题33 (2015年上半年上午试题48)

对高级语言源程序进行编译或解释的过程可以分为多个阶段，解释方式不包含__(48)__阶段。

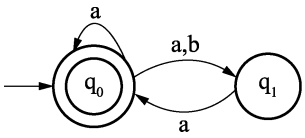
(48) A．词法分析　　B．语法分析　　C．语义分析　　D．目标代码生成

答案：D

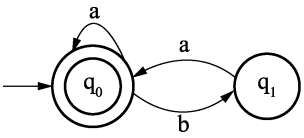
解析：解释方式运行用户程序时，直接执行源程序或者源程序的中间表示形式，不产生源程序的目标程序。

试题34 (2015年上半年上午试题49)

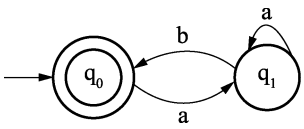
某非确定的有限自动机(NFA)的状态转换如下图所示(q_0既是初态也是终态)，与该 NFA 等价的确定的有限自动机(DFA)是__(49)__。



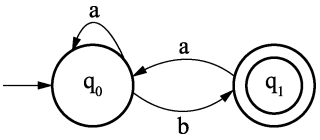
(49) A.



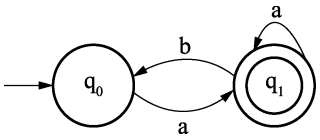
B.



C.



D.



答案：A

解析：一个非确定自动机(NFA)在读入符号串之后，并不确切地知道自动机处于哪个状态。但可以肯定一定处于状态集中的某一状态。该状态集记做$\{q_1,q_2,\cdots,q_k\}$。而一个等价的确定自动机(DFA)读入同样的 w 一定处于某个确定的状态。这样，都是读入同样的 w，DFA

到达某一个状态，而 NFA 到达某一个状态集。由 w 的任意性，可将 NFA 的所有状态集和 DFA 的状态一一对应起来。这种对应的前提就是能识别同样的输入串。

试题 35　(2015 年上半年上午试题 50)

递归下降分析方法是一种__(50)__方法。

(50) A．自底向上的语法分析　　B．自上而下的语法分析
　　C．自底向上的词法分析　　D．自上而下的词法分析

答案：B

解析：自顶向下的语法分析方法包括递归下降分析法和预测分析法。递归下降分析法的基本思想是：为每一个非终结符构造一个子程序，每个子程序的过程体按该产生式候选选项分情况展开，遇到终结符即进行匹配，而遇到非终结符则调用相应的子程序。

2.4　强化训练

2.4.1　综合知识试题

试题 1

属于面向对象、解释型程序设计语言的是__(1)__。

(1)　A．XML　　B．Python　　C．Prolog　　D．C++

试题 2

算术表达式“(a-b)*(c+d)”的后缀式是__(2)__。

(2)　A．ab-cd+*　　B．abcd-*+　　C．ab-*cd+　　D．ab-c+d*

试题 3

将高级语言源程序翻译成机器语言程序的过程，常引入中间代码。以下关于中间代码的叙述中，不正确的是__(3)__。

(3)　A．中间代码不依赖于具体的机器
　　B．使用中间代码可提高编译程序的可移植性
　　C．中间代码可以用树或图表示
　　D．中间代码可以用栈和队列表示

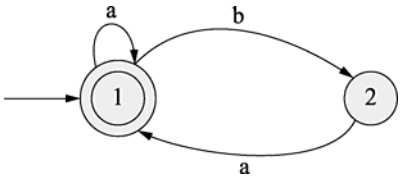
试题 4

对高级语言源程序进行编译的过程可以分为多个阶段，分配寄存器的工作在__(4)__阶段进行。

(4)　A．词法分析　　B．语法分析　　C．语义分析　　D．目标代码生成

试题 5

以下关于下图所示有限自动机的叙述中，不正确的是__(5)__。



(5) A．该自动机识别的字符串中 a 不能连续出现
B．自动机识别的字符串中 b 不能连续出现
C．自动机识别的非空字符串必须以 a 结尾
D．自动机识别的字符串可以为空串

试题 6

对于大多数通用程序设计语言，用__(6)__描述其语法即可。

(6) A．正规文法　　B．上下文无关文法
C．上下文有关文法　　D．短语结构文法

试题 7

以下程序设计语言中，__(7)__更适合用来进行动态网页处理。

(7) A．HTML　　B．LISP　　C．PHP　　D．JAVA/C++

试题 8

在引用调用方式下进行函数调用是将__(8)__。

(8) A．实参的值传递给形参　　B．实参的地址传递给形参
C．形参的值传递给实参　　D．形参的地址传递给实参

试题 9

编译程序对高级语言源程序进行编译的过程中，要不断收集、记录和使用源程序中一些相关符号的类型和特征等信息，并将其存入__(9)__中。

(9) A．符号表　　B．哈希表　　C．动态查找表　　D．栈和队列

试题 10

以下关于实现高级程序设计语言的编译和解释方式的叙述中，正确的是__(10)__。

(10) A．在编译方式下产生源程序的目标程序，在解释方式下不产生
B．在解释方式下产生源程序的目标程序，在编译方式下不产生
C．编译和解释方式都产生源程序的目标程序，差别是优化效率不同
D．编译和解释方式都不产生源程序的目标程序，差别在是否优化

试题 11

大多数程序设计语言的语法规则用__(11)__描述即可。

(11) A．正规文法　　B．上下文无关文法　　C．上下文有关文法　　D．短语结构文法

试题 12

在某 C/C++程序中，整型变量 a 的值为 0 且应用在表达式“c=b/a”中，则最可能发生的情形是__(12)__。

(12) A．编译时报告有语法错误　　B．编译时报告有逻辑错误
　　C．运行时报告有语法错误　　D．运行时产生异常

2.4.2　综合知识试题参考答案

【试题 1】

参考答案：B

要点解析：

XML 是一种用于标记电子文件使其具有结构性的标记语言。

C++是在 C 语言的基础上发展起来的，主要增加了类的功能，使其成为面向对象的程序设计语言。

Prolog 是以特殊的逻辑推理形式回答用户的查询，经常用于数据库和专家系统。

Python 是一种面向对象的解释型计算机程序设计语言。Python 语法简洁而清晰，具有丰富和强大的类库。它能够把用其他语言(尤其是 C/C++)制作的各种模块很轻松地连接在一起。

【试题 2】

参考答案：A

要点解析：后缀式是波兰逻辑学家卢卡西维奇发明的一种表达方式，把运算符号写在运算对象的后面，例如把 a+b 写成 ab+，这种表示法的优点是根据运算对象和算符的出现次序进行计算，不需要使用括号。

【试题 3】

参考答案：D

要点解析：中间代码是源程序的一种内部表示，或称中间语言。中间代码的作用是可使编译程序的结构在逻辑上更为简单明确，使用中间代码可提高编译程序的可移植性，常见的有逆波兰记号、四元式、三元式和树。

【试题 4】

参考答案：D

要点解析：目标代码生成阶段应考虑直接影响目标代码速度的三个问题：一是如何生成较短的目标代码；二是如何充分利用计算机中的寄存器，减少目标代码访问存储单元的次数；三是如何充分利用计算机指令系统的特点，以提高目标代码的质量。

【试题 5】

参考答案：A

要点解析：图中 a 可代表两个步骤：状态 1→1，状态 2→1。如果两个 a 连续出现，则无法区分。

【试题 6】

参考答案：B

要点解析：上下文无关文法：形式语言理论中一种重要的变换文法，用来描述上下文无关语言，在乔姆斯基分层中称为 2 型文法，由于程序设计语言的语法基本上都是上下文无关文法，因此应用十分广泛。

【试题 7】

参考答案：C

要点解析：HTML 用于处理静态网页；LISP 是一种基于λ演算的函数式编程语言。PHP

是一种通用开源脚本语言，语法吸收了 C 语言、Java 和 Perl 的特点，利于学习，使用广泛，主要适用于 Web 开发领域，可以比 CGI 或者 Perl 更快速地执行动态网页。用 PHP 做出的动态页面与其他的编程语言相比，PHP 是将程序嵌入 HTML(标准通用标记语言下的一个应用)文档中去执行，执行效率比完全生成 HTML 标记的 CGI 要高许多；PHP 还可以执行编译后代码，可以加密和优化代码运行，使代码运行更快。

Java 是一种可以撰写跨平台应用软件的面向对象的程序设计语言。Java 技术具有卓越的通用性、高效性、平台移植性和安全性，广泛应用于个人 PC、数据中心、游戏控制台、科学超级计算机、移动电话和互联网。

C++是一个接近系统底层的综合的、支持面向对象和范编程的程序设计语言，适用于开发要求很高的程序，例如大型游戏、大型企业应用、系统应用等。

【试题 8】

参考答案：B

要点解析：引用调用是把实参(如 int a)的地址(&a)赋给形参(指针变量，比如*b，这时 b=&a，即 b 指向变量 a)，如果*b(也即 a 对应的内存空间)发生变化，也就是变量 a 的值发生了变化。

【试题 9】

参考答案：A

要点解析：编译过程中，编译程序不断汇集和反复查证出现在源程序中各种名字的属性和特征信息等有关信息。这些信息通常记录在一张或几张符号表中。符号表的每一项有两部分：一部分是名字(标识符)；一部分是名字属性(标识符的有关信息)。编译过程中，每当扫描器(词法分析器)识别出一个名字后，编译程序就查阅符号表，看其是否在符号表中。符号表在编译全过程的地位和作用非常重要，是进行上下文合法性检查和语义处理及代码生成的依据。符号表总体结构的设计和实现与源语言的复杂性(包括词法结构、语法结构的复杂性)有关，还与对编译系统在时间效率和空间效率方面的要求有关。

【试题 10】

参考答案：A

要点解析：在编译方式下，机器上运行的是与源程序等价的目标程序，源程序和编译程序都不再参与目标程序的执行过程；而在解释方式下，解释程序和源程序要参与到程序的运行过程中，运行程序的控制权在解释程序。解释器翻译源程序时不产生独立的目标程序，而编译器则需要将源程序翻译成独立的目标程序。

【试题 11】

参考答案：B

要点解析：形式语言理论中有一种重要的变换文法，用来描述上下文无关语言，在乔姆斯基分层中称为 2 型文法。由于程序设计语言的语法基本上都是上下文无关文法，因此应用十分广泛。上下文无关文法拥有足够强的表达力来表示大多数程序设计语言的语法。另一方面，上下文无关文法又足够简单，使得我们可以构造有效的分析算法来检验一个给定字串是否由某个上下文无关文法产生。

【试题 12】

参考答案：D

要点解析：编译时 a 的值无法确定，表达式“c=b/a”符合 C/C++语言的语法逻辑，编译时不会报错。运行时，代入 a 的值，发生错误。